Josh Fann

Median-of-Medians Parameterized Selection Algorithm Report

I modified the P1driver.java to use a random array of integers and find the median of that array. To find an optimal colSize and colMed to use in practice, I ran a large sized array of 5000 integers into the algorithm looking for the median. I had the program return the number of comparisons as I iterated over each colSize from 3→10 and a colMed from 1→colSize. Below are the results of the number of comparisons each pair of values took to return the median:

colSize	colMed	Comparisons
3	1	105453
3	2	34830
3	3	96833
4	1	105070
4	2	33413
4	3	32711
4	4	95180
5	1	110376
5	2	36817
5	3	30306
5	4	36332
5	5	110176
6	1	124915
6	2	43261
6	3	32800
6	4	31958
6	5	41448
6	6	117821
7	1	146552
7	2	50896
7	3	36317
7	4	32873
7	5	36157
7	6	50540
7	7	140473
8	1	162628
8	2	61286
8	3	42028
8	4	36331
8	5	35945
8	6	43072
8	7	60076
8	8	161468
9	1	176401
9	2	67674
9	3	46824
9	4	40046
9	5	40070
9	6	41712
9	7	48054
9	8	67014
9	9	172780
10	1	192656
10	2	74253
10	3	54194
10	4	45547
10	5	40500
10	6	40417
10	7	44873
10	8	52794
10	9	72756
10	10	190218

The optimal choice for choosing which colSize and which colMed to choose is (5, 3), respectively. This is what we expect to generally happen, which means my algorithm is working like it should (thankfully). Another interesting note is that the # of comparisons is least for each colSize when colMed = colSize/2. And as you go further from colSize/2, the # of comparisons seems to follow an increasing parabola shaped curve. This is to my best guess because we are “guessing” where the median might be and when it finds that value, it has to partition the array. If the colMed we use is very high or very low, when partitioning, the data we “sieve” will not be as efficient as if we were to use a colMed closer to colSize/2.